



DEPARTMENT OF INFORMATION
AND COMMUNICATION
ENGINEERING

TP 11

VUEJS, NODEJS, CRUD APIs, MONGOOSE

INTERNET
PROGRAMMING

Submitted to: lecturer HOK TIN

e20190146
Chhorn Kakada
GIC-A

Tuesday, 13. June. 2023. 11:11PM

Github Account:

<https://github.com/ChhornKakada/IntenetProgramming/tree/tp11>

Noted: in this tp11, I only done with API want do it with **NestJs**.

CRUD categories

1. Add new

Cannot create new category, if the user not yet login.

The screenshot shows the Postman interface. In the left sidebar, under 'Collections', there is a 'New Collection' section containing 'JWT authentication', 'Ecommerce bookstore', and 'Ip Tp11 - 127.0.0.1:3002'. Under 'Ip Tp11 - 127.0.0.1:3002', there is a 'CRUD category' section with several methods listed: 'POST /new', 'GET /list', 'GET /id', 'POST /update/:id', 'POST /delete/:id', and 'GET /subcategory/list/:ctgId'. The 'POST /new' method is selected. The 'Body' tab shows a JSON payload:

```

1
2   ...
3     "name": "Phone",
4     ...
5       "description": "a device that uses either a system of wires along which electrical signals are sent or a system of radio signals to make it possible for you to speak to someone in another place who has a similar device: Just then, his phone rang."

```

The response body shows an error message:

```

1
2   "success": false,
3   "error": "You must sign in!"

```

Create successfully when user already logged in.

The screenshot shows the Postman interface. In the left sidebar, under 'Collections', there is a 'New Collection' section containing 'JWT authentication', 'Ecommerce bookstore', and 'Ip Tp11 - 127.0.0.1:3002'. Under 'Ip Tp11 - 127.0.0.1:3002', there is a 'CRUD category' section with several methods listed: 'POST /new', 'GET /list', 'GET /id', 'POST /update/:id', 'POST /delete/:id', and 'GET /subcategory/list/:ctgId'. The 'POST /auth/login' method is selected. The 'Body' tab shows a JSON payload:

```

1
2   ...
3     "username": "sacda",
4     ...
5       "password": "123456"

```

The response body shows a successful response:

```

1
2   "success": true,
3   "user": {
4     ...
5       "id": "6485a78043d8c75c22ff2b68",
6       "firstname": "Kakada",
7       "lastname": "Chhorn",
8       "email": "chhoenkakada1@gmail.com",
9       "username": "sacda",
10      "password": "$2a$10$EZPV1MT7q0nxxZxkkDk8G.0dCeCBicszL5FMBLJytIcezGnve8oN0",
11      "createdAt": "2023-06-11T10:52:48.607Z",
12      "updatedAt": "2023-06-12T11:07:48.135Z",
13      ...
14    }

```

POST /category/new

```

1   ...
2   ...
3   ...
4   ...
5   ...
6   ...
7   ...
8   ...
9   ...
10  ...
11  ...

```

Body (Pretty) Response:

```

1   "status": true,
2   "data": [
3     {
4       "_id": "64888742dc9874fa06ff0fb",
5       "name": "Phone",
6       "description": "a device that uses either a system of wires along which electrical signals are sent or a system of radio signals to make it possible for you to speak to someone in another place who has a similar device: Just then, his phone rang.",
7       "createdAt": "2023-06-13T15:12:02.365Z",
8       "updatedAt": "2023-06-13T15:12:02.365Z",
9       "__v": 0
10    }
11  ]

```

2. List all

This route list all category even if the user login or not.

GET /category/list

```

1   ...
2   ...
3   ...
4   ...
5   ...
6   ...
7   ...
8   ...
9   ...
10  ...
11  ...
12  ...
13  ...
14  ...
15  ...
16  ...
17  ...
18  ...
19  ...
20  ...

```

Body (Pretty) Response:

```

1   "status": true,
2   "data": [
3     {
4       "_id": "64888742dc9874fa06ff0fb",
5       "name": "Phone",
6       "description": "a device that uses either a system of wires along which electrical signals are sent or a system of radio signals to make it possible for you to speak to someone in another place who has a similar device: Just then, his phone rang.",
7       "createdAt": "2023-06-13T15:12:02.365Z",
8       "updatedAt": "2023-06-13T15:12:02.365Z",
9       "__v": 0
10    },
11    {
12      "_id": "648887c1dc9874fa06ff0fbe",
13      "name": "Computer",
14      "description": "a machine or device that performs processes, calculations and operations based on instructions provided by a software or hardware program.",
15      "createdAt": "2023-06-13T15:14:09.505Z",
16      "updatedAt": "2023-06-13T15:14:09.505Z",
17      "__v": 0
18    }
19  ]

```

3. Find by id

When the user input wrong id.

The screenshot shows the Postman interface with the following details:

- Collections:** IP Tp11 - 127.0.0.1:3002
- APIs:** CRUD category
- Method:** GET /id
- URL:** http://127.0.0.1:3002/category/648810c8bc3cc3e1b63e6646
- Body:** (Pretty) JSON response:


```

1 "status": false,
2 "message": "Category with this ID is not found."
3
4 
```
- Headers:** (7)
- Test Results:** 200 OK, 8 ms, 299 B

When the user input the correct id.

The screenshot shows the Postman interface with the following details:

- Collections:** IP Tp11 - 127.0.0.1:3002
- APIs:** CRUD category
- Method:** GET /id
- URL:** http://127.0.0.1:3002/category/648887c1dc9874fa06ff0fbe
- Body:** (Pretty) JSON response:


```

1 "status": true,
2 "data": {
3   "_id": "648887c1dc9874fa06ff0fbe",
4   "name": "Computer",
5   "description": "a machine or device that performs processes, calculations and operations based on instructions provided by a software or hardware program.",
6   "createdAt": "2023-06-13T15:14:09.505Z",
7   "updatedAt": "2023-06-13T15:14:09.505Z",
8   "__v": 0
9 }
10 
```
- Headers:** (7)
- Test Results:** 200 OK, 7 ms, 553 B

4. Update by id

The user need to login before deleting any category.

The screenshot shows the Postman interface with the following details:

- Collections:** IP Tp11 - 127.0.0.1:3002
- APIs:** CRUD category
- Method:** POST /update/:id
- URL:** http://127.0.0.1:3002/category/update/648810c8bc3cc3e1b63e6645
- Body:** (Pretty) JSON response:


```

1 "description": "asda123asdfasdfl123"
2
3 
```
- Headers:** (9)
- Test Results:** 200 OK, 6 ms, 280 B

The user can update only data that they wise. Example below the user just want to update only description field of the category. But if you want to update other fields you can mention it in the body.

POST /category/update/648887c1dc9874fa06ff0fbe

```

1 "description": "test updating"

```

Body Cookies (1) Headers (7) Test Results
Pretty Raw Preview Visualize JSON

```

1 "status": true,
2 "message": "Updated successfully!",
3 "data": {
4     "_id": "648887c1dc9874fa06ff0fbe",
5     "name": "Computer",
6     "description": "test updating",
7     "createdAt": "2023-06-13T15:14:09.505Z",
8     "updatedAt": "2023-06-13T15:22:24.897Z",
9     "__v": 0
10 }
11
12

```

5. Delete by id

The delete route is also need to login before update too. When the user input the wrong id, it will response that the id is wrong, otherwise the category with that id will be deleted and return its category.

POST /category/delete/648813b75f2ff8444ea042a3

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description	...	

Body Cookies (1) Headers (7) Test Results
Pretty Raw Preview Visualize JSON

```

1 "status": false,
2 "message": "Category with this ID is not found."

```

The screenshot shows the Postman application interface. On the left, the 'Scratch Pad' sidebar lists various collections and environments. The main workspace displays a POST request to 'http://127.0.0.1:3002/category/delete/648887c1dc9874fa06ff0fbe'. The 'Body' tab shows a JSON response with status: true, message: "Deleted successfully!", and data containing the deleted category details. The status bar at the bottom indicates a 201 Created response.

6. List all subcategories by categoryId

The output will display the list of all the category that defined by categoryId.

The screenshot shows the Postman application interface. The 'Scratch Pad' sidebar lists various collections and environments. The main workspace displays a GET request to 'http://127.0.0.1:3002/category/subcategory/list/64888742dc9874fa06ff0fb'. The 'Body' tab shows a JSON response with status: true and data: [...]. The data array contains multiple subcategory objects, each with an _id, name, and description. The status bar at the bottom indicates a 200 OK response.

When the user input wrong ID of category.

The screenshot shows the Postman application interface. In the left sidebar, under 'Collections', there is a 'New Collection' section with items like 'JWT authentication', 'Ecommerce bookstore', and 'IpTp11 - 127.0.0.1:3002'. Under 'IpTp11 - 127.0.0.1:3002', there is a 'CRUD category' collection containing 'POST /new', 'GET /list', 'GET /id', 'POST /update/:id', and 'POST /delete/:id'. The main workspace shows a 'GET' request to 'http://127.0.0.1:3002/category/subcategory/list/64888742dc9874fa06df0fb'. The response status is 200 OK, and the response body is a JSON object: { "status": true, "data": [] }.

7. Apply middleware in the category controllers

```

1 import { MiddlewareConsumer, Module, NestModule, RequestMethod } from '@nestjs/common';
2 import { CategoryController } from './category.controller';
3 import { CategoryService } from './category.service';
4 import { CategorySchema } from './schemas/category.schema';
5 import { MongooseModule } from '@nestjs/mongoose';
6 import { EnsureSignedInMiddleware } from 'src/auth/middlewares/ensure-signed-in.middleware';
7
8 @Module({
9   imports: [MongooseModule.forFeature([{ name: 'Category', schema: CategorySchema }])],
10  controllers: [CategoryController],
11  providers: [CategoryService]
12 })
13 export class CategoryModule implements NestModule {
14   configure(consumer: MiddlewareConsumer) {
15     consumer
16       .apply(EnsureSignedInMiddleware)
17       .forRoutes(
18         { path: 'category/new', method: RequestMethod.POST },
19         { path: 'category/delete/:id', method: RequestMethod.POST },
20         { path: 'category/update/:id', method: RequestMethod.POST }
21       );
22     }
23   }
24 }
```

CRUD subcategories

1. Add new

In this route you need to login to create a new subcategory (item).

The screenshot shows the Postman interface. On the left, the 'Collections' sidebar is open, showing various API endpoints under 'Ip Tp11 - 127.0.0.1:3002'. The 'CRUD subcategory' collection is selected, and the 'POST subcategory/new' endpoint is highlighted. In the main workspace, a POST request is being made to `http://127.0.0.1:3002/subcategory/new`. The 'Body' tab is selected, showing the following JSON payload:

```

1 ... "name": "Iphone",
2 ... "description": "a smartphone made by Apple that combines a computer, iPod, digital camera and
3 ... cellular phone into one device with a touchscreen interface.",
4 ... "categoryId": "64888742dc9874fa06ff0fb"
5 ...

```

The response section shows a 201 Created status with a response time of 8 ms and a size of 596 B. The response body is also displayed in JSON format:

```

1 "status": true,
2 "data": {
3     "name": "Iphone",
4     "description": "a smartphone made by Apple that combines a computer, iPod, digital camera
5         and cellular phone into one device with a touchscreen interface.",
6     "categoryId": "64888742dc9874fa06ff0fb",
7     "_id": "64888b90dc9874fa06ff0fc",
8     "createdAt": "2023-06-13T15:30:24.137Z",
9     "updatedAt": "2023-06-13T15:30:24.137Z",
10    "__v": 0
11 }
12

```

2. List all

List all category even if the user is logged in or not.

The screenshot shows the Postman interface. The 'Collections' sidebar is open, and the 'CRUD subcategory' collection is selected. The 'GET subcategory/list' endpoint is highlighted. A GET request is being made to `http://127.0.0.1:3002/subcategory/list`. The 'Body' tab is selected, showing the following JSON response:

```

1 "status": true,
2 "data": [
3     {
4         "_id": "64888b90dc9874fa06ff0fc",
5         "name": "Iphone",
6         "description": "a smartphone made by Apple that combines a computer, iPod, digital
7             camera and cellular phone into one device with a touchscreen interface.",
8         "categoryId": "64888742dc9874fa06ff0fb",
9         "createdAt": "2023-06-13T15:30:24.137Z",
10        "updatedAt": "2023-06-13T15:30:24.137Z",
11        "__v": 0
12    },
13    {
14        "_id": "64888c30dc9874fa06ff0fc",
15        "name": "Samsung",
16        "description": "is Samsung Electronics' flagship line of Android smartphones and
17            tablets.",
18        "imgUrl": "https://encrypted-tbn0.gstatic.com/images?
19            q=tbn:ANd9GcSZB0mwg0Zwla4-BVqj7mXsislq2P0PtxZK05l0w0KrBvcIqPFieMPoi1w3LhI62F3HMc",
20        "categoryId": "64888742dc9874fa06ff0fb",
21        "createdAt": "2023-06-13T15:33:04.029Z",
22        "updatedAt": "2023-06-13T15:33:04.029Z",
23        "__v": 0
24    }
25 ]

```

3. Find by id

It will tell the user if the user has input the wrong ID, otherwise return the data.

The screenshot shows a Postman collection for an API. The 'GET subcategory/id' request is selected. The URL is `http://127.0.0.1:3002/subcategory/64882fcefdc4b19dc62cfcbd`. The response body is:

```

1
2 "status": false,
3 "message": "Subcategory with this ID is not found."
4

```

The screenshot shows the same Postman collection. The 'GET subcategory/id' request is selected. The URL is `http://127.0.0.1:3002/subcategory/64888c30dc9874fa06ff0fcf`. The response body is:

```

1
2 "status": true,
3 "data": {
4     "_id": "64888c30dc9874fa06ff0fcf",
5     "name": "Samsung",
6     "description": "is Samsung Electronics' flagship line of Android smartphones and tablets.",
7     "imgUrl": "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSZBQmWgOZWo1A4-BYqJ7mXsislq2P0PtxZKD5l0w0KiBvcIqPFieMPoi1w3LhI62F3HMc",
8     "categoryId": "64888742dc9874fa06ff0fb",
9     "createdAt": "2023-06-13T15:33:04.029Z",
10    "updatedAt": "2023-06-13T15:33:04.029Z",
11    "__v": 0
12 }
13

```

4. Update by id

It will tell user if the user input the wrong ID, otherwise update the data.

The screenshot shows the same Postman collection. The 'POST subcategory/update/id' request is selected. The URL is `http://127.0.0.1:3002/subcategory/update/64882fcefdc4b19dc62cfcbd`. The response body is:

```

1
2 {"name": "Samsung test 1",
3 "description": "demo changeing description"
4

```

The 'Body' tab shows the raw JSON payload:

```

1
2 {
3     "name": "Samsung test 1",
4     "description": "demo changeing description"
5

```

The screenshot shows the Postman interface with the following details:

- Collections:** IP Tp11 - 127.0.0.1:3002
- Method:** POST
- URL:** http://127.0.0.1:3002/subcategory/update/64888c30dc9874fa06ff0fcf
- Body (JSON):**

```

1 ...
2   ...
3     "name": "Samsung test 1",
4     "description": "demo changeing description"
      
```
- Response Headers:** 201 Created, 7 ms, 661 B
- Response Body (Pretty JSON):**

```

1   "status": true,
2   "message": "Updated successfully!",
3   "data": {
4     "_id": "64888c30dc9874fa06ff0fcf",
5     "name": "Samsung test 1",
6     "description": "demo changeing description",
7     "imgUrl": "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcsZBQmWgOZWo1A4-BYqJ7mXsislqP0PtxZKD5l0w0KiBvcIqPFieMPoi1w3LhI62F3HMc",
8     "categoryId": "64888742dc9874fa06ff0fb",
9     "createdAt": "2023-06-13T15:33:04.029Z",
10    "updatedAt": "2023-06-13T15:46:34.572Z",
11    "__v": 0
12  }
13
14
      
```

5. Delete by id

It will tell the user that user the ID is wrong when the user input the wrong ID. Otherwise delete it and return its data.

The screenshot shows the Postman interface with the following details:

- Collections:** IP Tp11 - 127.0.0.1:3002
- Method:** POST
- URL:** http://127.0.0.1:3002/subcategory/delete/64882fcfdcb19dc62cfcb
- Body (JSON):**

```

1 ...
2   ...
3     "status": false,
4     "message": "Subcategory with this ID is not found."
      
```
- Response Headers:** 201 Created, 8 ms, 307 B

The screenshot shows the Postman interface with the following details:

- Collections:** IP Tp11 - 127.0.0.1:3002
- Method:** POST
- URL:** http://127.0.0.1:3002/subcategory/delete/64888c30dc9874fa06ff0fcf
- Body (JSON):**

```

1 ...
2   ...
3     "status": true,
4     "message": "Deleted successfully!",
5     "data": {
6       "_id": "64888c30dc9874fa06ff0fcf",
7       "name": "Samsung test 1",
8       "description": "demo changeing description",
9       "imgUrl": "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcsZBQmWgOZWo1A4-BYqJ7mXsislqP0PtxZKD5l0w0KiBvcIqPFieMPoi1w3LhI62F3HMc",
10      "categoryId": "64888742dc9874fa06ff0fb",
11      "createdAt": "2023-06-13T15:33:04.029Z",
12      "updatedAt": "2023-06-13T15:46:34.572Z",
13      "__v": 0
14    }
      
```
- Response Headers:** 201 Created, 8 ms, 661 B

6. List all products by productId

The screenshot shows the Postman interface. On the left, there's a sidebar with sections like Collections, APIs, Environments, Mock Servers, Monitors, and History. Under Collections, there's a tree view of API endpoints. The main area shows a GET request to `http://127.0.0.1:3002/subcategory/product/list/64888b90dc9874fa06ff0fc`. The response body is displayed in a JSON prettified format:

```

1 "status": true,
2 "data": [
3     {
4         "_id": "64888b90dc9874fa06ff0fc",
5         "name": "Iphone",
6         "description": "a smartphone made by Apple that combines a computer, iPod, digital camera and cellular phone into one device with a touchscreen interface.",
7         "items": [
8             {
9                 "_id": "648890c0dc9874fa06ff0fd",
10                "name": "Iphone 13 pro max",
11                "description": "use an Apple-designed A15 Bionic processor featuring a 16-core neural engine, 6-core CPU (with 2 performance cores and 4 efficiency cores), and 5-core GPU.",
12                "imgUrl": "https://i.pinimg.com/736x/8e/e6/c6/8ee6c665954cb903cd2c6a5f5b763c01.jpg"
13            },
14            {
15                "_id": "64889182dc9874fa06ff0fe",
16                "name": "iPhone 14",
17                "description": "The iPhone 14 display has rounded corners that follow a beautiful curved design, and these corners are within a standard rectangle.",
18                "imgUrl": "https://i.pinimg.com/564x/26/be/56/26be56634ad9773c9d8f6315cac2cba7.jpg"
19            }
20        ]
21    }
22 ]
23
24
25
26

```

7. Implement the middleware

```

1 import { MiddlewareConsumer, Module, NestModule, RequestMethod } from '@nestjs/common';
2 import { SubcategoryController } from './subcategory.controller';
3 import { SubcategoryService } from './subcategory.service';
4 import { MongooseModule } from '@nestjs/mongoose';
5 import { SubcategorySchema } from './schemas/subcategory.schema';
6 import { EnsureSignedInMiddleware } from 'src/auth/middlewares/ensure-signed-in.middleware';
7
8 @Module({
9     imports: [MongooseModule.forFeature([{ name: 'Subcategory', schema: SubcategorySchema }])],
10    controllers: [SubcategoryController],
11    providers: [SubcategoryService]
12 })
13
14 // user middleware here
15 export class SubcategoryModule implements NestModule {
16     configure(consumer: MiddlewareConsumer) {
17         consumer
18             .apply(EnsureSignedInMiddleware)
19             .forRoutes(
20                 { path: "/subcategory/new", method: RequestMethod.POST },
21                 { path: "/subcategory/update/:id", method: RequestMethod.POST },
22                 { path: "/subcategory/delete/:id", method: RequestMethod.POST }
23             )
24     }
25 }
26

```

CRUD products

1. Add new

POST /product/new

```

1 "name": "iPhone 14",
2     "description": "The iPhone 14 display has rounded corners that follow a beautiful curved
3         design, and these corners are within a standard rectangle.",
4             "categoryId": "64888742dc9874fa06ff0fb",
5                 "subctgId": "64888b90dc9874fa06ff0fc",
6                     "imgUrl": "https://i.pinimg.com/564x/26/be/56/26be56634ad9773c9d8f6315cac2cba7.jpg"
7
8
9
10
11
12
13
14

```

Pretty Raw Preview Visualize JSON

Body Cookies (1) Headers (7) Test Results

201 Created 11 ms 713 B Save Response

status: true,
data: {
 name: "iPhone 14",
 description: "The iPhone 14 display has rounded corners that follow a beautiful curved
 design, and these corners are within a standard rectangle.",
 categoryId: "64888742dc9874fa06ff0fb",
 subctgId: "64888b90dc9874fa06ff0fc",
 _id: "64889182dc9874fa06ff0fe0",
 createdAt: "2023-06-13T15:55:46.701Z",
 updatedAt: "2023-06-13T15:55:46.701Z",
 __v: 0
}

2. List all

List all products.

GET /product/list

Pretty Raw Preview Visualize JSON

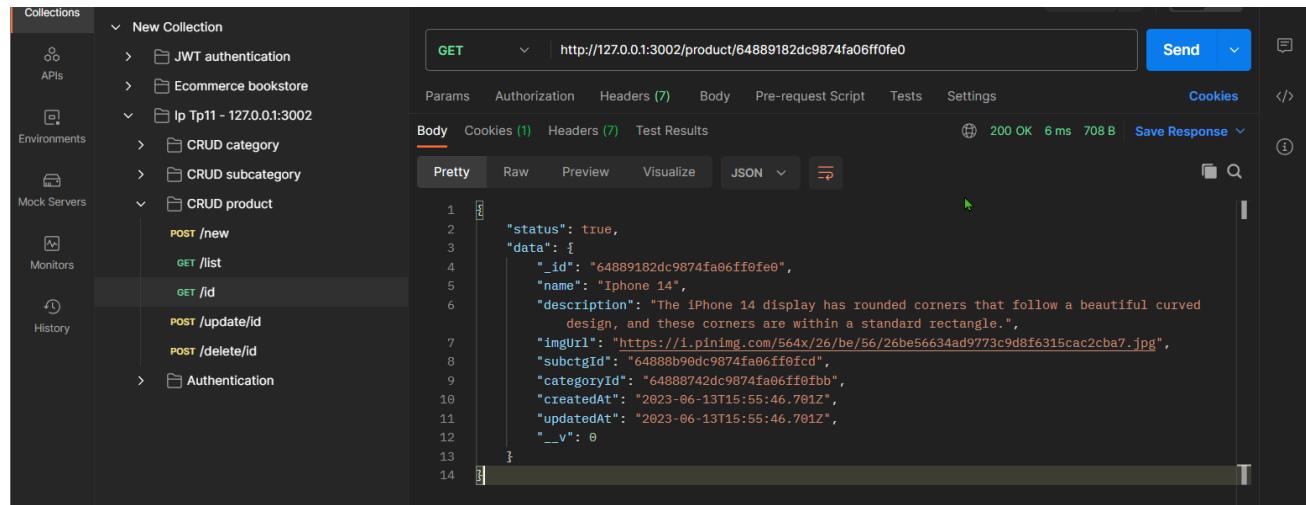
Body Cookies (1) Headers (7) Test Results

200 OK 8 ms 1.17 KB Save Response

status: true,
data: [
 {_id: "648890c0dc9874fa06ff0fd",
 name: "iPhone 13 pro max",
 description: "use an Apple-designed A15 Bionic processor featuring a 16-core neural
 engine, 6-core CPU (with 2 performance cores and 4 efficiency cores), and 5-core
 GPU.",
 imgUrl: "https://i.pinimg.com/736x/8e/e6/c6/8ee6c665954cb993cd2c6a5f5b763c01.jpg",
 subctgId: "64888b90dc9874fa06ff0fc",
 categoryId: "64888742dc9874fa06ff0fb",
 createdAt: "2023-06-13T15:52:32.089Z",
 updatedAt: "2023-06-13T15:52:32.089Z",
 __v: 0},
 {_id: "64889182dc9874fa06ff0fe0",
 name: "iPhone 14",
 description: "The iPhone 14 display has rounded corners that follow a beautiful
 curved design, and these corners are within a standard rectangle.",
 imgUrl: "https://i.pinimg.com/564x/26/be/56/26be56634ad9773c9d8f6315cac2cba7.jpg",
 subctgId: "64888b90dc9874fa06ff0fc",
 categoryId: "64888742dc9874fa06ff0fb",
 createdAt: "2023-06-13T15:55:46.701Z",
 updatedAt: "2023-06-13T15:55:46.701Z",
 __v: 0}
]

3. Find by id

It will show the data of the product if the user input the correct ID. Otherwise, it will return the message “Product with this ID is not found”.

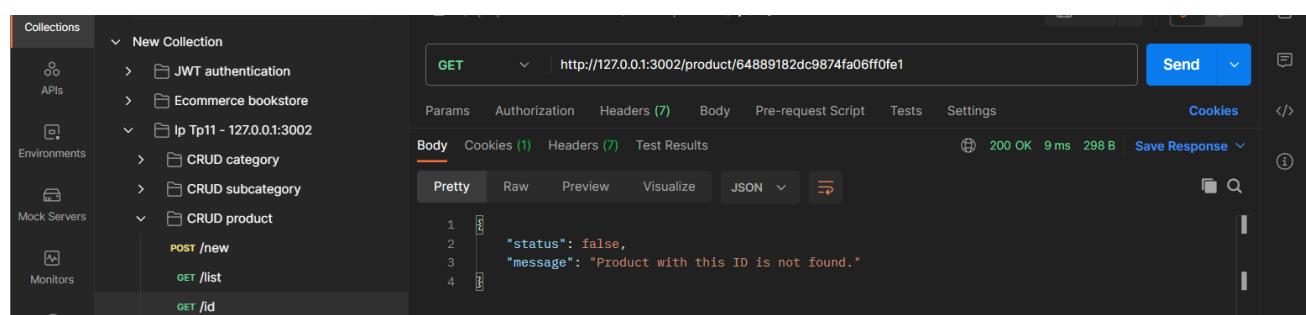


The screenshot shows the Postman interface with a successful API call. The URL is `http://127.0.0.1:3002/product/64889182dc9874fa06ff0fe0`. The response status is 200 OK, and the response body is:

```

1 "status": true,
2 "data": {
3     "_id": "64889182dc9874fa06ff0fe0",
4     "name": "Iphone 14",
5     "description": "The iPhone 14 display has rounded corners that follow a beautiful curved design, and these corners are within a standard rectangle.",
6     "imgUrl": "https://i.pinimg.com/564x/26/be/56be56634ad9773c9d8f6315cac2cba7.jpg",
7     "subctgId": "64888b90dc9874fa06ff0fc0",
8     "categoryId": "64888742dc9874fa06ff0fb0",
9     "createdAt": "2023-06-13T15:55:46.701Z",
10    "updatedAt": "2023-06-13T15:55:46.701Z",
11    "__v": 0
12 }
13
14

```



The screenshot shows the Postman interface with an unsuccessful API call. The URL is `http://127.0.0.1:3002/product/64889182dc9874fa06ff0fe1`. The response status is 200 OK, and the response body is:

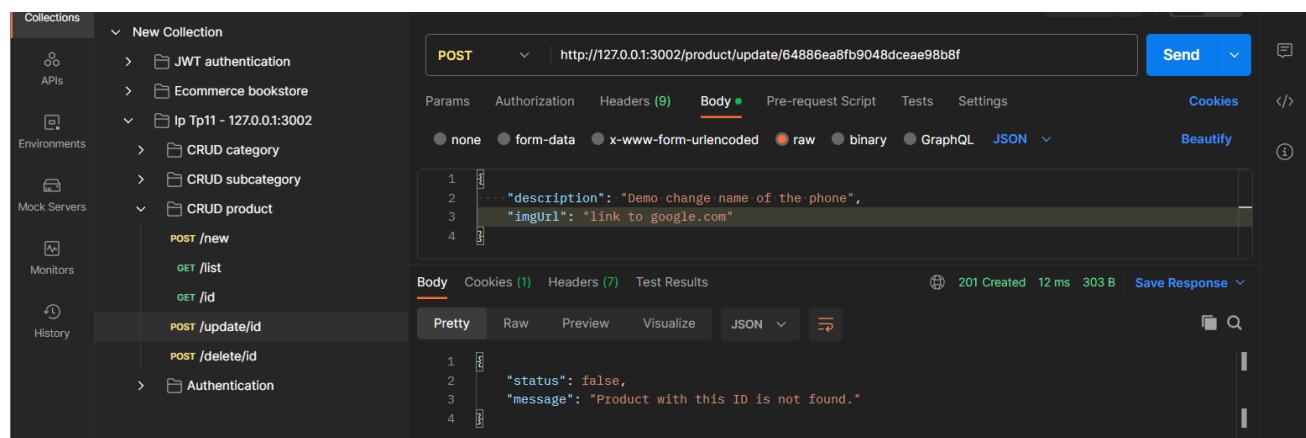
```

1 "status": false,
2 "message": "Product with this ID is not found."
3
4

```

4. Update by id

It will update if the ID is correct, otherwise it will send the message that cannot found.



The screenshot shows the Postman interface with an unsuccessful API call. The URL is `http://127.0.0.1:3002/product/update/64886ea8fb9048dceae98b8f`. The response status is 201 Created, and the response body is:

```

1 "status": false,
2 "message": "Product with this ID is not found."
3
4

```

5. Delete by id

It will delete the product if the ID is correct, otherwise return with the message “Product with this ID is not found”.

6. Implement the middleware

```
1 import { MiddlewareConsumer, Module, NestModule, RequestMethod } from '@nestjs/common';
2 import { ProductController } from './product.controller';
3 import { ProductService } from './product.service';
4 import { ProductSchema } from './schemas/product.schema';
5 import { MongooseModule } from '@nestjs/mongoose';
6 import { EnsureSignedInMiddleware } from 'src/auth/middlewares/ensure-signed-in.middleware';
7
8 @Module({
9   imports: [MongooseModule.forFeature([{ name: 'Product', schema: ProductSchema }])],
10  controllers: [ProductController],
11  providers: [ProductService]
12 })
13
14
15 // user middleware here
16 export class ProductModule implements NestModule {
17   configure(consumer: MiddlewareConsumer) {
18     consumer
19       .apply(EnsureSignedInMiddleware)
20       .forRoutes(
21         {path: "product/new", method: RequestMethod.POST},
22         {path: "product/delete/:id", method: RequestMethod.POST},
23         {path: "product/update/:id", method: RequestMethod.POST}
24       )
25   }
26 }
27
```